시간 지연 신경망을 이용한 음악 장르 분류

이재원[†] · 조찬윤[††] · 김상균[†††]

요 약

본 논문에서는 오디오 데이터의 효과적인 검색을 위하여, 시간 지연 신경망을 이용한 음악 장르 분류 시스템을 제안한다. 분류 대상 장르는 Blues, Country, Hard Core, Hard Rock, Jazz, R&B(Soul), Techno, Trash Metal의 8종류이다. 장르를 분류하기 위한 비교단위는 곡 중에서의 한 마디이다. 이러한 마디는 리듬의 특성을 효과적으로 반영하는 스네어 드럼 소리를 기준으로 추출한다. 분류기는 시간 지연 신경망을 이용하여 구성하며 입력은 추출된 마디에 대한 주파수 특징벡터이다. 제안한 시스템의 유효성을 검증하기 위한 실험에서, 장르별 10곡씩 총 80곡의 학습 데이터와 장르별 5곡씩 총 40곡의 테스트 데이터에 대하여 각각 92.5%와 60%의 정인식율을 보였다.

Music Genre Classification using Time Delay Neural Network

Jae-Won Lee[†], Chan-Yun Cho[††] and Sang-Kyoon Kim[†††]

ABSTRACT

This paper proposes a classifier of music genre using time delay neural network(TDNN) for an audio data retrieval systems. The classifier considers eight kinds of genres such as Blues, Country, Hard Core, Hard Rock, Jazz, R&B(Soul), Techno and Trash Metal. The comparative unit to classify the genres is a melody between bars. The melody pattern is extracted based on snare drum sound which represents the periodicity of rhythm effectively. The classifier is constructed with the TDNN and uses fourier transformed feature vector of the melody as input pattern. We experimented the classifier on eighty training data from ten musics for each genres and forty test data from five musics for each genres, and obtained correct classification rates of 92.5% and 60%, respectively.

1. 서 론

최근 컴퓨팅 환경의 발달과 인터넷의 보편화로 인하여 다양한 응용분야에서 멀티미디어 데이터의 사용이 급증하고 있다. 따라서 멀티미디어 데이터에 대한 사용자들의 검색 요구도 증가하고 있으며, 이를 위한 검색 시스템에 대한 연구가 활발히 진행되고 있다. 그러나 이미지나 동영상 데이터에 대해서는 다양한 검색 방법론이 제시되고 있으나, 오디오 데이터에 대한 연구는 드물다.

인터넷상에서 오디오 데이터 서비스를 제공하는 업체들은 수작업에 의존하고 있다. 즉 전문가가 모든 곡들에 대해 해당 장르, 분위기, 빠르기등의 다양한 정보를 수작업으로 추출하고, 이를 데이터베이스에 저장하여 사용자 질의에 사용한다. 이러한 검색시스템은 새로운 곡들을 추가할 때마다 수작업으로 곡에 대한 정보를 추출해야하므로 많은 정보화 비용이 요구된다.

내용 기반의 선행연구들은 대부분 사용자가 곡의 제목을 알지 못할 때, 제목 대신 다른 방법으로 곡을 검색할 수 있게 하는 방법에 대한 연구이다. 주로 허

본 연구는 한국과학재단 목적기초연구(2001-1-51200-007-1) 지원으로 수행되었음.

[†] 인제대학교 전산학과 대학원 석사과정
[††] 한국통신데이터 마케팅본부 전임연구원
[†††] 정회원, 인제대학교 정보컴퓨터공학부 조교수

밍이나 사용자가 알고있는 계명을 질의어로 사용하여 오디오 데이터베이스에 저장된 곡을 검색한다 [1-3]. 이러한 방법에서는 사전에 모든 곡들에 대해 허밍이나 계명들과 같이 비교할 수 있는 특징들을 추출하여 데이터베이스를 구축해야한다. 특히, 검색 시 곡 전체에 대한 특징이나 계명을 추출하여 비교할 수 없으므로 사용자가 많이 사용할 것으로 예상되는 부분을 추출해서 저장해야한다. 그러나 오디오 데이터의 방대함과 모호함으로 인하여 효과적인 특징 추출이 어렵고 또한 특징 비교 방법을 개발하기 힘들다. 따라서 효율적인 오디오 검색 시스템을 개발하기 위해서는 오디오 매칭 알고리즘과 무엇보다도 오디오 데이터로부터 관심 있는 영역의 추출 방법에 관한 연구가 선행되어야 한다.

본 논문에서는 시간 지연 신경망(TDNN)을 이용한 음악 장르 분류 시스템을 제안한다. 오디오 데이터의 영역 추출 방법론으로 곡 중에서의 마디 추출 방법과 오디오 데이터 비교 방법으로는 추출된 마디를 이용한 TDNN 분류기를 제시한다. 이러한 방법들을 적용하여 오디오 데이터를 효율적으로 처리 및 관리할 수 있는 음악 장르 분류 시스템을 제안한다.

음악은 멜로디와 리듬으로 양분할 수 있다. 연주 시 악기들은 마디에 근거하여 리듬을 표현한다. 리듬은 규칙성을 가지게 되며 드럼이나 베이스들이 주로 사용되는 리듬악기이다. 따라서 오디오 데이터에서 관심있는 영역의 기본 단위인 마디를, 음악의 리듬 규칙성에 근거하여 추출 할 수 있다. 본 논문에서는 드럼 중에서도 가장 주기적이며 음색이 뚜렷한 스네어 드럼 소리를 기준으로 마디를 추출한다.

분류기로 사용하는 TDNN은 다층 신경망에 동적 요소인 시간 지연 요소를 첨가하여 학습패턴의 동적인 특성을 위치에 무관하게 가미하도록 구성된 신경망이다. 학습시간은 길지만 실행 시 처리속도가 빠르다는 특성을 가지고 있기 때문에 오디오 데이터와 같이 데이터 양이 방대하고 많은 처리가 요구되는 검색기법에 효과적이다.

2. 음악 장르 분석 및 분류 대상 장르

현재 세상에는 매우 많고 다양한 종류의 음악 장르가 존재한다. 그리고 지금도 새로운 장르가 지속적으로 만들어지고 있으며, 또 어떤 장르는 다른 장르로 대체되어 소멸되기도 한다. 일반적으로 음악 장르는 어떤 음악 스타일이 유행하면서 만들어지고, 그 스타일이 고정화되면서 한 음악 장르로 자리 잡는다. 일부는 고정화되지 못하고 소멸되기도 한다. 그리고 이러한 음악 스타일이란 음악적인 면만을 일컫는 것은 아니다. 문화적 상황에 의해 노래 가사가 담고 있는 내용이나, 뮤지션들의 행위 등, 다양한 요소를 포함하는 의미이다. 이처럼 음악 장르는 다양한 요인들에 의해 만들어진다. 즉 뮤지션들의 새로운 시도에 의해 만들어지기도 하고, 상업적인 목적에 의해 만들어지기도 한다. 이러한 음악 장르들은 최근 두 장르 이상의 요소를 포함하여 크로스오버(cross-over)라는 새로운 장르가 만들어지거나, 또 어떤 뮤지션들의 앨범은 두 가지 이상의 음악 장르에 포함되기도 하는 것처럼 매우 복잡하다. 이처럼 현재 알려져 있는 음악 장르들은 특정한 기준에 의해 만들어진 것이 아니기 때문에, 어떤 특정한 기준을 통해 분류하기 힘들다.

본 논문에서는 이러한 대중음악 특성을 고려하여 음악 장르 중에서 일반화되어 고정화된 음악 장르를 선별하고, 이들 장르 중 내용, 즉 가사가 담고 있는 의미나 어떤 문화적인 요인에 의해 만들어지지 않은, 순수한 음악적 특성을 가진 음악 장르들을 분류하고자 하는 대상 음악 장르로 선택하였다. 즉 특정한 리듬이나 음악적인 스타일이 정해져 있고, 악기 구성 또한 거의 일정한 음악 장르를 선택하였다. 이러한 규칙에 의해 선택된 음악 장르는 Blues, Country, Hard Core, Hard Rock, Jazz, R&B(Soul), Techno, Trash Metal의 8개 음악 장르이다. 실험에 사용할 데이터들은 각 음악 장르에 대해 수집한 인터넷 자료와 관련 참고도서에서 공통적으로 가장 해당 장르를 잘 표현하며 충실하다고 평가되는 뮤지션들의 앨범들에서 선별하여 구성하였다[4,5].

3. 음악 장르 분류 시스템

본 논문에서 제시하는 분류 시스템의 전체 구성은 그림 1과 같다. 전처리 단계에서는 스테레오 사운드를 모노로 변환한다. 마디 시작점 찾기 단계에서는 스네어 드럼의 특성을 이용하여 마디 시작 위치와 길이를 구한다. 마디 추출 단계에서는 특징 추출을 위한 한 마디를 추출한다. 데이터 정규화 단계에서는 다운샘플링을 통하여 추출된 데이터의 크기를 정규

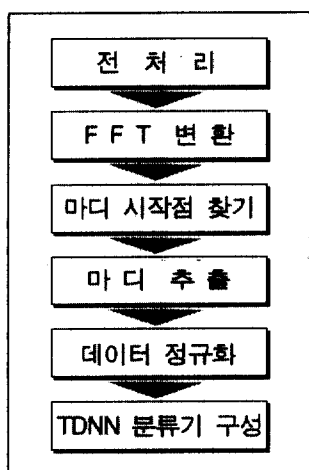



그림 1. 분류 시스템의 구성 단계

화한다. FFT 단계에서는 추출된 마디에서 FFT를 통하여 특징을 추출하고, 이를 학습패턴으로 구성한다. 마지막으로 TDNN 분류기 구성 단계에서는 구성된 학습패턴으로 학습하여 음악 장르 분류 시스템을 구성한다.

3.1 전처리

CD 음질의 웨이브 파일은 일반적으로 44.1 KHz, 16bit로 샘플링 된다. 그리고 일반적으로 4분 정도의 웨이브 파일의 크기는 약 40MByte 정도이며, 스테레오로 녹음되어 있으므로 2개의 출력 채널을 가진다. 전처리 단계에서는 이와 같이 방대한 데이터의 연산 회수를 줄이기 위해 스테레오 사운드를 모노로 변환한다.

3.2 마디 시작점 찾기 및 마디 추출

일반적으로 모든 음악은 박자와 빠르기라는 요소를 가진다. 그리고 곡마다 이러한 박자와 빠르기는 다르기 때문에 모든 곡에 대해 같은 위치에서 같은 크기를 가져오는 것은 의미가 없다. 따라서 본 논문에서는 마디의 시작점들을 찾고, 특징 추출에 사용하기 위한 크기로 곡 중에서 한 마디의 크기를 추출한다. 실험에서 다른 대부분의 데이터들을 분석해 본 결과, 대부분의 곡들이 4/4 박자 또는 6/8 박자이며, 빠르기도 일정하다. 그리고 대부분의 곡들은 멜로디를 담당하는 부분과 리듬을 담당하는 부분으로 구성되어 있다. 그리고 멜로디를 구성하는 부분보다 리듬을 구성하는 부분에서 규칙성을 가진다. 본 연구에서는 마디를 찾기 위해 이러한 리듬 부분의 특성에 의해 마디의 시작점을 찾는다.

리듬은 대부분 드럼과 베이스 기타라는 악기 또는 이와 유사한 기능을 하는 악기를 사용하여 연주되는 특성을 가지고 있다. 예를 들면, 째즈에서는 베이스 기타 대신 이와 유사한 더블 베이스를 사용하던지, 드럼 대신에 여러 다른 타악기를 사용한다. 그러나 이러한 다른 악기로 연주된 곡이라 할지라도 대부분 비슷한 음역을 표현한다. 본 논문에서 마디의 시작점을 찾기 위해 드럼파트를 구성하고 있는 스네어 드럼이라는 악기에 초점을 맞추었다. 이 악기는 대부분 곡의 리듬에서 2번째와 4번째 박자에 규칙적으로 나타나며, 표현되는 주파수가 거의 일정하다는 특성을 가진다.

일반적으로 곡들은 뮤지션들이 의도한 곡의 분위기를 만들기 위하여 도입부분을 가지고 있다. 이러한 도입부분들은 대부분 특별한 음악장르의 특성을 갖지 않는다. 그리고 곡에 따라 템포도 자유로운 경우가 많기 때문에 도입부분을 제거해야 한다. 본 논문의 실험에 사용된 데이터들을 조사해 본 결과 곡들의 도입부분의 크기가 40초 미만이었다. 따라서 이러한 도입부분을 제거하기 위해 웨이브 파일의 앞부분 40초가 지난 다음부터 곡들의 마디 시작점을 찾는다. 마디의 시작점을 찾기 위해 일반적인 마디의 크기가 220,000개의 샘플링 이하라는 특징을 고려하여 40초가 지난 다음부터 262,144개의 샘플을 채취한다. 그림 2는 데이터 추출의 예를 보여준다.

그리고, 스네어 악기의 소리를 찾기 위해, 시간축상에서의 신호를 주파수 영역에서의 신호로 변환하여 음성 분석이나 음향 분석에 널리 사용되는 방법인 FFT를 사용한다. 본 논문에서는 256-point FFT를 1,024번 수행하였으며, 푸리에 변환은 대칭성을 가지므로 1번째에서 128번째까지의 주파수대역만 사용

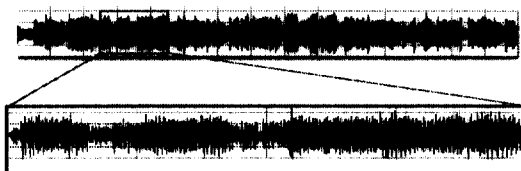



그림 2. 마디 추출을 위한 데이터 추출의 예

하였다. 따라서 변환된 데이터는 1024×128의 크기를 가지며, 1024는 시간을 나타내고, 128은 주파수대역폭을 나타낸다.

변환된 데이터를 분석해 본 결과, 모든 데이터에서 스네어 악기의 소리는 37번째 주파수 대역에서부터 43번째 주파수 대역 사이에서 잘 나타나는 사실을 알 수 있었다. 그림 3은 실험 데이터에서 37번째 주파수의 예이며, 가장 높은 두 개의 정점이 한 마디를 간격으로 한 스네어 드럼 부분이다.

FFT를 사용하여 변환된 값들에 대해 아래의 순서(Step 1~Step 7)에 의해 마디의 시작점과 크기를 구한다.

Step 1: 스네어 악기의 소리 후보를 찾기 위해 1024의 시간을 32씩의 32개의 구간으로 나누어, 37번째에서 43번째까지 각 주파수 대역 별로 각 구간 최대값을 찾는다. 그림 4는 각 구간에서의 최대값을 찾은 결과이다.

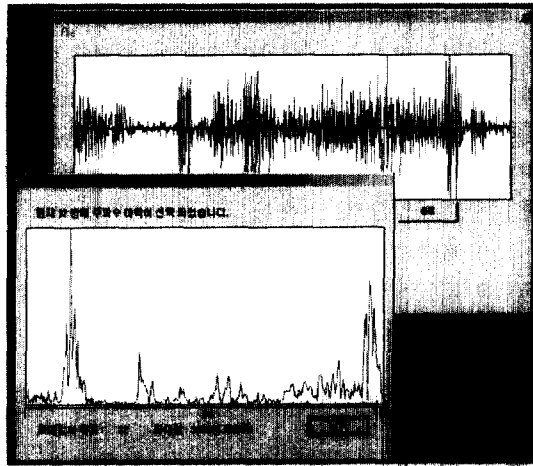



그림 3. 37번째 주파수의 예

37번째 주파수		38번째 주파수		39번째 주파수		40번째 주파수		41번째 주파수	
29	58726.41613	29	68017.96394	28	30614.37407	26	31826.09291	31	38691.63671
32	49269.9499	35	46711.51763	36	39461.74501	37	36548.07153	33	38034.16366
72	30925.68459	75	34545.4045	71	27159.09117	73	20865.62443	78	25406.41743
125	155507.6651	122	99884.17477	120	136116.0941	120	132857.4673	124	203678.5793
130	71765.04678	134	59182.08837	132	72316.63399	131	50447.54609	129	193379.8583
162	33615.89026	169	24521.01937	161	24236.24871	163	30173.3788	161	51265.21682
218	33605.72914	214	70556.98193	205	41426.06854	212	28701.37431	217	46396.89862
230	28411.28547	254	34647.3736	252	24540.63256	253	43815.97191	256	39807.17614
260	35528.51553	256	25454.55891	259	39249.16178	264	27727.88844	287	45520.99373
301	139338.2976	299	95421.05396	300	115732.9775	305	79808.08092	303	226106.7328
342	54894.08728	350	51355.31158	342	54512.41289	346	61650.11623	340	44381.22683
402	62088.14274	403	70294.69962	405	82667.33304	399	99019.80081	403	62742.08742
426	78264.43302	425	67414.48015	438	60322.82679	422	53460.81593	421	46334.13314
479	76176.82538	479	90825.95906	479	90120.82448	454	39536.63388	470	67173.07374
480	96054.37992	491	91472.45388	485	87970.65529	482	118056.4524	486	244321.3329
534	21278.83416	533	22649.12146	517	32473.35123	532	28314.90633	516	43177.98783
573	87401.52281	572	58973.3961	572	59221.61311	573	62060.54246	573	57701.97641
579	58380.07791	579	39405.46723	579	29009.41614	576	45466.96042	577	66218.98912
617	25936.08557	623	24105.88955	613	15614.14444	619	25468.33685	615	28093.47996
660	103189.206	661	69034.80363	665	90469.59686	665	195091.0029	666	304375.242
677	91067.35451	672	89290.50625	675	49205.17617	676	42513.7115	673	95948.32318
707	52271.80772	706	24456.85615	707	26096.46619	717	21136.07767	717	27484.49863
753	66558.11509	756	54918.10538	752	65464.34048	755	57263.1007	755	58993.94264
770	23552.05149	769	17838.89377	770	19399.9078	769	18864.04928	770	25164.89192
805	24168.86482	804	20064.0437	800	19918.61152	802	19066.7908	800	25275.57010
845	173067.628	846	119740.2169	851	88256.60842	850	139434.637	852	240393.2074
864	55885.09311	864	41467.39983	892	37035.69284	864	39296.459	869	68880.09491
896	21439.44994	901	27701.08826	918	13951.02355	899	25203.96323	901	22234.7631
936	62285.43439	942	45913.60929	945	42887.7444	935	54224.63857	947	41006.49112
982	43959.40463	990	27635.18439	984	17871.19656	980	41517.00463	980	37926.40871
1023	40372.00986	1023	94415.45288	1023	80381.97396	1023	45952.56081	1023	58491.8011

그림 4. (Step 1)의 예

Step 2: 위에서 구한 구간별 최대값을 정렬하여 각 주파수에서 상위 12개의 값들을 후보로 선정한다. 그림 5는 [Step 2]의 예이다.

37번째 주파수		38번째 주파수		39번째 주파수		40번째 주파수	
845	173067.628	846	119740.2169	120	136116.0941	665	195091.0028
125	155507.6651	122	99884.17477	300	115732.9775	850	139434.637
301	139338.2976	299	95421.05396	665	90469.59686	120	132857.4673
660	103189.206	1023	94415.45288	479	90120.82448	482	118056.4524
480	96054.37992	491	91472.45388	851	88256.60842	399	99019.80081
677	91067.35451	479	90825.95906	485	87970.65529	305	79808.08092
573	87401.52281	672	89290.50625	405	82667.33304	573	62060.54246
426	78264.43302	214	70556.98193	1023	80381.97396	346	61650.11623
479	76176.82538	403	70294.69962	132	72316.63399	755	57263.1007
130	71765.04678	661	69034.80363	752	65464.34048	935	54224.63857
753	66558.11509	29	68017.96394	438	60322.82679	422	53460.81593
936	62285.43439	425	67414.48015	572	59221.61311	131	50447.54609
402	62088.14274	134	59182.08837	342	54512.41289	1023	45952.56081
579	58380.07791	572	58973.3961	675	49205.17617	576	45466.96042
864	55885.09311	756	54918.10538	945	42887.7444	253	43815.97191
342	54894.08728	360	52332.66097	205	41426.06854	676	42513.7115
707	52271.80772	350	51355.31158	36	39461.74501	980	41517.00463
29	50726.41613	35	46711.51763	259	39249.16178	454	39536.63388
32	49269.9499	942	45913.60929	892	37035.69284	864	39296.459
982	43959.40463	864	41467.39983	517	32473.35123	37	36548.07153
1023	40372.00986	579	39405.46723	28	30614.37407	26	31826.09291
352	37595.69605	254	34647.3736	579	29009.41614	163	30173.3788
260	35528.51553	75	34545.4045	71	27159.09117	212	28701.37431
162	33615.89026	901	27701.08826	353	26342.35336	532	28314.90633
218	33605.72914	990	27635.18439	707	26096.46619	264	27727.88844
72	30925.68459	256	25454.55891	252	24540.63256	619	25468.33685
230	28411.28547	169	24521.01937	161	24236.24871	899	25203.96323
617	25936.08557	706	24456.85615	800	19918.61152	356	22571.67466
805	24168.86482	623	24105.88955	770	19399.9078	717	21136.07767
770	23552.05149	533	22649.12146	984	17871.19656	73	20865.62443
896	21439.44994	804	20064.0437	613	15614.14444	802	19066.7908
534	21278.83416	769	17838.89377	918	13951.02355	769	18864.04928

그림 5. (Step 2)의 예

Step 3: 각 주파수에서 선정된 값들 중 시간 위치 값이 32 간격 보다 작은 후보들을 비교하여 값이 낮은 쪽을 후보에서 삭제한다.

37번째 주파수

845 : 173067.628
125 : 155507.6651
301 : 139434.637
660 : 103189.206
480 : 96054.37992
677 : 91067.35451
573 : 87401.52281
426 : 78264.43302
479 : 76176.82538
130 : 71765.04678
753 : 66558.119509
936 : 62285.43439

삭제후 남은 37번째 주파수 후보

845 : 173067.628
125 : 155507.6551
301 : 139434.637
660 : 103189.206
480 : 96054.37992
573 : 87401.52281
426 : 78264.43302
753 : 66558.119509
936 : 62285.43439

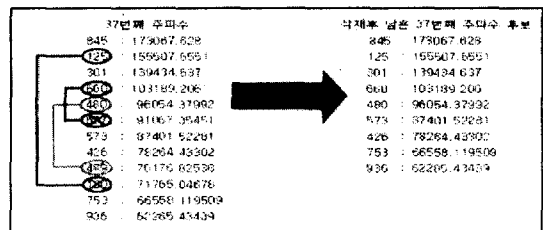



그림 6. (Step 3)의 예

Step 4: 각 주파수별로 남겨진 후보들을 다른 주파수의 후보들과 비교하여, 시간 위치 차이가 ±10 이하인 위치 값들을 찾아서 투표한 후, 위치 값들의 평균값을 구한다.

37 주파수 후보위치값	38 주파수 후보위치값	39 주파수 후보위치값	40 주파수 후보위치값	41 주파수 후보위치값	42 주파수 후보위치값	43 주파수 후보위치값
845	846	120	665	666	122	483
125	122	300	850	486	488	667
660	299	665	120	852	669	125
480	1023	479	482	303	303	851
573	491	851	399	124	850	303
426	672	405	305	577	573	406
753	214	1023	573	403	424	212
936	403	752	346	755	346	251
	29	438	755			
		572	935			

1. 평균값 : (845 + 846 + 851 + 850 + 852 + 850 + 851) / 7 = 849
   투표 개수 : 7
2. 평균값 : (125 + 122 + 120 + 120 + 124 + 122 + 125) / 7 = 123
   투표 개수 : 7

⋮

그림 7. (Step 4)의 예

Step 5: 투표 결과가 6 이상인 후보들만을 추출한 후 시간 순서대로 나열한다.

849, 123, 302, 665, 483 => 123, 302, 483, 665, 849

그림 8. [Step 5]의 예

Step 6: 시간 순으로 나열된 후보들 사이의 시간 간격을 구한 후, 이들 중 가장 큰 값과 가장 작은 값을 뺀 나머지 값들의 평균을 구한다. 그리고 한 마디의 크기를 찾기 위해 시간 간격을 2배한다.

시간 간격 : 179, 181, 182, 184
시간 간격 평균 : 181.5 => 182 (반올림)
한 마디의 크기 : 364 × 256

그림 9. [Step 6]의 예

Step 7: 마디 시작점은 최종 후보 값들 중 가장 큰 값을 선정하고, [Step 6]에서 구한 시간 간격 평균을 2로 나누어 뺀 값이다.

만약 후보 값들 중 123의 값이 가장 크다면 :
123 - 91 = 32
실제 샘플링 위치값 : 32 × 256

그림 10. [Step 7]의 예

위의 순서에 의해 찾아진 시작점과 마디의 길이는 256-point FFT에 의한 시간 간격이므로 실제 데이터의 시작점과 마디의 길이는 256을 곱한 값이다. 따라서 구해진 마디 시작점과 크기에 256을 곱한 후 한 마디의 크기만큼의 데이터를 추출한다. 그림 11은 구현된 마디 찾기 프로그램이다.

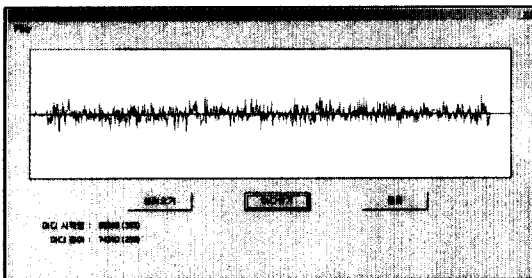



그림 11. 마디 추출 프로그램

3.3 데이터 정규화

획득된 데이터들은 그 곡의 빠르기에 따라 한 마디의 크기가 다르기 때문에 추출되는 특징값들의 개수가 다르다는 문제가 발생한다. 그러므로 모든 데이터에서 동일한 크기의 특징을 추출하기 위해 데이터를 정규화하는 과정이 필요하다. 본 논문에서는 데이터를 정규화하기 위해 앞에서 구한 한 마디의 샘플링 길이를 이용하여 다운샘플링하였다. 다운샘플링 값은 다음 식에 의해 계산한다.

$$X = \frac{44100 \times 16384}{L}$$

(X : 다운샘플링 값, L : 추출된 마디의 길이) (1)

위 식에 의해 각 데이터들은 16384개의 샘플을 가진 wave 파일로 변환된다.

3.4 학습 패턴 생성

시간 지연 신경망의 입력으로 사용될 학습 패턴을 구성하기 위해 다운샘플링으로 크기가 정규화 된 데이터에 대하여 256-point FFT를 64번 수행한다. 이를 통해 64×256개의 변환된 값들을 획득한다. 이렇게 변환된 값들 중 주파수 대역으로 1번째부터 128번째까지 128개에 대해 순서대로 4개씩 뽑아내어 이를 합하여 32개의 값을 추출하고, 이를 64번 반복하여 64×32 패턴을 만든다. 추출된 패턴을 신경망의 학습 패턴으로 사용하기 위해 이를 0과 1사이의 값으로 정규화한다. 그림 12는 생성된 학습 패턴의 예이다.

그림 12. 추출된 학습 패턴의 예

3.5 TDNN 분류기 구성

TDNN은 정적구조 신경망에 동적 요소(delay, integration)를 첨가하여 음성과 같이 동적인 특성을 가진 데이터에 대해 인식할 때 좋은 인식률을 가지기 때문에 음성 인식 분야에서 많이 사용되고 있다 [6-10].

전체적으로 TDNN은 입력층(input layer), 2개의 은닉층(hidden layer), 출력층(output layer)의 다층으로 구성된다. 은닉층에서는 시간 지연 요소를 통한 입력으로 음악의 국부적인 특징을 감지해내고 출력층에서는 전단 은닉층의 시간적인 지연을 갖는 출력의 제곱을 더하여 출력을 한다. 이렇게 해서 최종적으로 은닉층에서는 음악이 가지는 국부적인 특성을 감지함으로써 패턴을 시각적으로 굴곡(time-warping)하게 되고, 출력층에서는 전단의 시간적으로 지연된 출력들을 합함으로써 입력패턴에 지연현상이 발생하여도 같은 출력을 낼 수 있는 특징을 갖는다.

일반적인 TDNN의 전체적인 구조는 그림 13, 14와 같다[6-10]. 그림 14에서 (a)는 전체적인 구조를 나타내고, (b)는 TDNN을 옆에서 본 모습을 나타낸다. 그리고 (b)의 Step Size는 시간 지연의 정도를 나타내는 요소이고, Kernal Size는 다음 층의 한 노드로 전달될 때, 계산되는 시간축의 노드의 개수를 나타낸다.

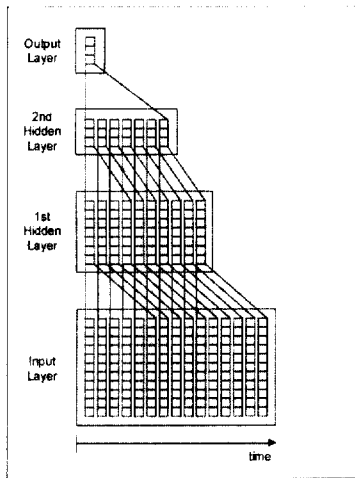



그림 13. 각 계층에 대한 전체 구조

(a) 전체적인 구조 (b) 옆에서 본 구조

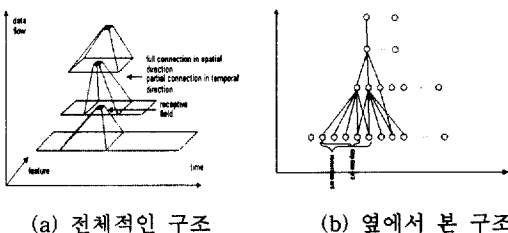



그림 14. TDNN 구조

다층구조 신경망에서 사용되는 일반적인 학습 알고리즘(learning algorithm)인 역전파(BP; Back-Propagation) 알고리즘의 목적함수(object function)는 다음과 같다.

$$E = \frac{1}{2}\sum_j (o_j - d_j)^2 \tag{2}$$

식 (2)에서 o_j는 j번째 출력노드 y_j의 활성화값(activation value)이고, 따라서 목적함수 E를 y_j에 대해 편미분하면 출력노드의 활성화값에 대한 오차의 편미분값을 얻게 된다.

$$\frac{\partial E}{\partial y_j} = y_j - d_j \tag{3}$$

역전파 학습 알고리즘에서는 이 값이 역전파에 대한 시작점이 된다. 그러나 TDNN에서 출력은 전단의 은닉층에서 지연된 출력값의 제곱을 더한 것이다.

$$o_j = \sum_t y_{jt}^2 \tag{4}$$

따라서 식 (4)를 식 (2)의 정의에 대입하여 시간 t에 대해 편미분하면 다음과 같다.

$$\frac{\partial E}{\partial y_{jt}} = 2y_{jt}((\sum_t y_{jt}^2) - d_j) \tag{5}$$

일반적인 역전파 신경망에서와 같이 오차의 편미분값을 얻게 되고, 이 값이 TDNN에서의 역전파 학습의 시작점이 된다.

오류 역전파 시 연결강도(weight)를 변화시키는데 있어서는 시간의 위치에 무관하게 입력의 특성들을 추출하기 위하여 시간 지연된 연결강도들의 변화값의 평균만큼을 변화시킨 다음 각 연결강도에 복사하게 된다. 그런데 TDNN은 연결강도와 출력노드가 매우 많기 때문에 학습 시 매우 많은 시간이 소요된다. 따라서 학습 시간을 단축하기 위해서 여러 고속화 알고리즘을 사용하게 되는데, 본 논문에서 학습시 오류를 급격히 줄이고 진동(oscillation)을 방지하기 위해 학습률(learning rate)과 모멘텀(momentum)을 오류의 변화 정도에 따라 변화시키기 위해 사용하였다.

제안한 음악 장르 분류시스템에서 사용한 TDNN 구조는 1개의 입력층, 2개의 은닉층, 그리고 1개의 출력층으로 일반적인 TDNN 구조와 동일하게 이루어져 있으며, 실제 사용한 각 층의 노드 구성은, 입력

층은 64×32개의 노드로, 은닉층은 각각 31×16, 10×8개의 노드로 구성하였으며, 마지막으로 출력층은 3개의 출력 노드를 가지도록 구성하였다. 그리고 각 층에 대하여 Kernal Size와 Step Size를 실험에 의하여 설정하였다. TDNN 분류기의 각 계층에 대한 세부 구성은 표 1과 같다.

표 1. 음악 장르 분류 시스템의 TDNN 구성

	Spatial Size	Kernal Size	Step Size	노드 개수
입력층	32	0	0	64×32
은닉층 1	16	4	2	31×16
은닉층 2	8	4	1	10×8
출력층	1	4	3	3×1

4. 실험 및 결과 분석

TDNN을 이용한 음악 장르 분류 시스템은 펜티엄 PC, Windows 환경에서 Visual C++로 구현하였다.

그림 15는 구성된 시간 지연 신경망을 학습하는 모습이다. 학습률은 0.05, 모멘텀은 0.5, 목표 에러값은 0.01, 제한 반복회수는 30,000번으로 지정하여 학습하였다.

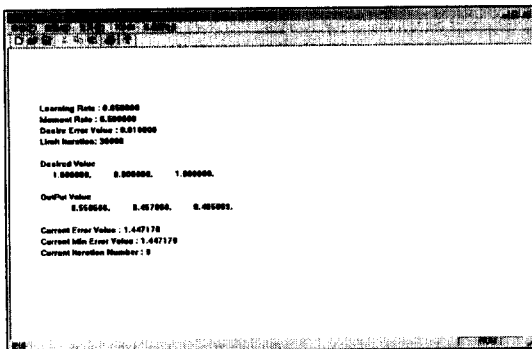



그림 15. TDNN의 학습

학습된 음악 장르 분류 시스템의 성능을 평가하기 위해, 수집된 총 120곡의 각 장르별 음악 데이터 중 학습에 사용된 데이터와 학습에 사용되지 않은 데이터로 구분하여 실험하였다. 학습 데이터는 각 장르별로 10개씩 총 80개를, 테스트 데이터는 장르별로 5개씩 총 40개를 사용하였다. 이러한 학습 데이터와 테스트 데이터는 객관적인 성능 평가를 위해 서로 다른 곡에서 각각 추출하였다. 그림 16은 학습된 TDNN 분류기로 테스트하는 모습이다.

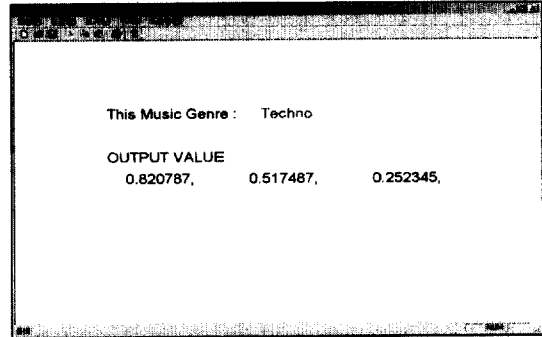



그림 16. 음악 장르 분류 시스템

실험 결과, 학습 데이터를 분류하였을 경우, 92.5%의 정인식률(正認識率)을 보였고, 테스트 데이터에 대해서는 60%의 정인식률을 보였다. 표 2는 실험 결과이다. 전체적으로 학습 데이터의 정인식률은 92.5%로 높은데 비하여, 테스트 데이터의 정인식률은 60%로 비교적 낮았다.

표 2. 실험 결과

음악 장르	학습 데이터		테스트 데이터	
	정인식	오인식	정인식	오인식
Blues	9	1	4	1
Country	9	1	2	3
Hard Core	8	2	2	3
Hard Rock	10	0	4	1
Jazz	10	0	4	1
R&B (Soul)	9	1	2	3
Techno	10	0	4	1
Trash Metal	9	1	2	3
총계	74	6	24	16
인식률	92.5%	7.5%	60%	40%

장르별로 오인식(誤認識)한 결과를 보면 Hard Core와 Trash Metal의 오인식율이 높았는데, Hard

Core는 Trash Metal로 오인식한 경우가 많았고, Trash Metal 또한 Hard Rock과 Hard Core로 오인식하였다. 이는 Hard Core와 Trash Metal이 모두 Hard Rock에서 파생된 음악 장르로 연주하는 악기 구성과 리듬이 거의 유사하기 때문이다. Blues, Jazz, R&B의 오인식 예도 이와 유사한 경우이다. 이러한 장르들은 일반인들도 분류하기 힘든 장르들이다.

그 외의 완전히 다른 장르들에서의 오인식 이유는 학습에 사용된 다른 데이터들과 리듬과 빠르기가 상당히 달랐기 때문이었다. 그리고 이러한 템포 차이는 마디 추출에도 영향을 끼쳐 대표 패턴의 생성에서부터 오류가 발생하게끔 하였다. 따라서 마디 추출 방법을 보완하여 각 장르의 곡에서 나타날 수 있는 다양한 템포를 수용할 수 있도록 하고, 이들을 효과적으로 반영하는 패턴들에 대하여 TDNN 분류기를 학습한다면 인식률을 향상시킬 수 있을 것이다. 표 3은 오인식한 데이터들에 대한 오인식한 음악 장르이다.

5. 결 론

본 논문에서는 TDNN을 이용한 음악 장르 분류 시스템을 제안하였다. 음악의 리듬 규칙성에 근거하여 마디를 장르 분류 단위로 사용하였다. 마디는 리듬 악기 중 가장 주기적이며 뚜렷한 음색을 가지는 스네어 드럼을 기준으로 추출하였다. 이러한 마디에 대한 FFT 특징 벡터를 대표 패턴으로 학습하여 TDNN 음악 장르 분류기를 구성하였다.

오디오 데이터의 내용기반 검색에 대한 연구가 미흡한 가운데 관심 있는 영역 추출 방법과 분류 방법에 대한 하나의 접근 방법을 제시하였다

제안한 시스템은 유사한 장르를 분류하기 위한 효과적인 특징 추출에 관한 연구, 다양한 리듬과 빠르

표 3. 오인식한 음악 장르

음악 장르	학습 데이터		테스트 데이터	
	곡 명	오인식한 장르	곡 명	오인식한 장르
Blues	Albert King "Answer to the laundromat blues"	Jazz	BBKing "I'm gonna do what they do to me"	R&B (Soul)
Country	Garth Brooks "American honky-tonk bar association"	Hard Rock	Bob Dylan "Tangled up in blue"	Hard Rock
			Garth Brooks "Cowboy Cadillac"	Hard Rock
			John Denver "Rocky mountain high"	Techno
Hard Core	Biohazard "Man with a promise"	Techno	Korn "Got the life"	Trash Metal
			Marilyn Manson "School drop-outs"	Trash Metal
	Nine Inch Nails "Mr self destruct"	Hard Rock	Rage Against The Machine "Born of a broken man"	Techno
Hard Rock	•	•	Deep Purple "Mistreated"	Hard Core
Jazz	•	•	Duke Ellington "Little max"	R&B (Soul)
R&B (Soul)	Maxwell "Sumthin' sumthin'"	Hard Rock	D' Angelo "Send it on"	Techno
			D' Angelo "The root"	Hard Rock
			Maxwell "The urban theme"	Jazz
Techno	•	•	Moby "Have you seen my baby"	R&B (Soul)
Trash Metal	Sepultura "Sympthom of the universe"	Hard Rock	Anthrax "In my world"	Hard Rock
			Metallica "Creeping death"	Hard Rock
			Slayer "Reborn"	Hard Core

기를 수용할 수 있는 마디 추출 방법의 보완, 다양한 템포의 곡들에 대한 대표 패턴 분류 방법에 대한 연구들을 통하여 성능을 개선 할 수 있을 것이다. 더불어 곡 중에서의 클라이맥스 추출 방법 개발, 효율적인 사용자 질의 및 비교 방법의 개발 등이 향후 연구 과제이다

참 고 문 헌

[1] Nako Kosugi, Yuichi Nishihara, Seiichi Kon'ya, Masashi Yamamuro and Kazuhiko Kushima, "Music Retrieval by Humming," *Proceedings of the 1999 IEEE Racific Rim Conference on Communications, Computers and Signal Processing*, pp. 404-407, 1999.

[2] S. Blackburn and D. DeRoure, "A Tool for Content-Based Navigation of Music," *Proceedings of the ACM Multimedia*, pp. 361-368, 1998.

[3] Chih-Chin Liu, Jia-Lien Hsu and Arbee L. P. Chen, "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval," *Proceedings of the IEEE Multimedia Systems*, pp. 451-456, 1999.

[4] 신현준, 록 음악의 아홉 가지 갈래들, 문학과지성사, ISBN 89-320-0969-4, 1997.

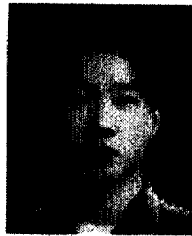
[5] 하세민, 볼륨을 높여라, 도서출판 꿈, ISBN 89-870-7201-0, 1996.

[6] 김동국, 정차균, 정홍, "한국어 음소 인식을 위한 시간 지연 신경망," 정보과학회논문지, 제18권, 제3호, pp. 300-312, 1991.

[7] 이영호, 정홍, "음절을 기반으로한 한국어 음성 인식," 전자공학회논문지, 제31권, 제1호, pp. 11-22,

[8] 박규봉, 이근배, 이종혁, "음소단위 TDNN에 기반한 한국어 연속 음성인식을 위한 데이터 자동 분할," 제7회 한글 및 한국어 정보처리학회지, pp. 30-34, 1995.

[9] 김경회, 이근배, 이종혁, "한국어 음성 언어 처리를 위한 음소 단위 인식과 형태소 분석의 결합," 정보과학회논문지, 제22권, 제10호, pp. 1488-1498

[10] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. on ASSP*, vol. 37, no. 3, 1989.

이 재 원

2000년 인제대학교 전산학과 졸업 (이학사)
2000년~현재 인제대학교 대학원 전산학과 석사과정
관심분야 : 정보검색, 패턴인식

조 찬 윤

1999년 인제대학교 전산학과 졸업 (이학사)
2001년 인제대학교 대학원 전산학과 졸업(이학석사)
2001년~현재 한국통신데이터 마케팅본부 전임연구원
관심분야 : 정보검색, 패턴인식

김 상 균

1991년 경북대학교 통계학과 졸업 (이학사)
1994년 경북대학교 대학원 컴퓨터공학과 졸업(공학석사)
1996년 경북대학교 대학원 컴퓨터공학과 졸업(공학박사)
1996년~현재 인제대학교 정보컴퓨터공학부 조교수
관심분야 : 정보검색, 정보보호, 패턴인식